TCP1101 Programming Fundamentals
Trimester 1, Session 2022/23 (T2215)
Faculty of Computing and Informatics
Multimedia University

Assignment (40%)
Alien vs. Zombie Game

1. Introduction

Morgan Stark is a child prodigy just like his father Tony Stark when he was young. At five years old, she self-taught herself C++ and mastered the concepts of structured programming and object-based programming.

Recently, she discovered a mobile game named *Alien Path* currently available in Android and iOS. In this game, the player navigates an alien on a path to destroy robots. It combines several elements of role-playing game, puzzle, path-planning, and strategy to provide a unique gaming experience.

Morgan intends to put her programming skills to the test by developing a simplified, text-based version of this game, with several modifications of her own. She names her modified game as *Alien vs. Zombie,* due to her peculiar soft spot for aliens and zombies.

2. Mission

This assignment aims to test your problem-solving skills using C++. You are to create a text-based, simplified version of the Alien Path game based on the given requirements, and submit the deliverables requested.

3. Deadlines

This assignment is divided into TWO (2) parts, each requires a submission and has a deadline:

A. Part 1 (10%)

Deadline: On or Before 11:59pm, 29 January 2023 (Sunday)

Submit the interim progress of your work. The source code must be compilable into an executable program that demonstrates the completed requirements.

B. Part 2 (30%)

Deadline: On or Before 11:59pm, 12 February 2023 (Sunday)

Submit the completed Deliverables. The game must be complete.

Refer to Deliverables for details about the submissions of each part.

4. Grouping

This assignment will be done in **a group of 2 to 3 students** from the **same tutorial section**. Students must **stick with the same group for Part 1 and Part 2**.

STRICTLY NO COPYING from other students in other groups or from any other sources (Internet, books, etc.). Plagiarism and cheating are serious offences in MMU, and you will fail in this subject and be reported to the faculty for action.

You may be called for an **INTERVIEW** to explain what you have done. Failure to prove your contributions to the assignment may lead to ZERO (0) mark for your assignment.

Each group member may be given different marks according to their contributions.

5. Getting Started

To help you get started, you are provided with the following:

A. Video Guides

A YouTube playlist that contains demos and guides to help you better understand the assignment requirements below. You will be notified if more guides are added to the playlist from time to time.

B. Starter Kit

A ZIP-compressed repository at GitHub that contains helper code and Markdown files (see Deliverables) to help you start coding and write documentations. Click the link to download the ZIP file and decompress it to see its content. Soon, we will guide you on how to install Git and create a GitHub account using your MMU Gmail account so that you can upload your assignments to GitHub.

6. Minimum Requirements

This section describes the minimum requirements of the Alien vs Zombie game.

6.1. Overview of Alien vs. Zombie

Alien vs Zombie is a turn-based combat game in which the player controls Alien to defeat a group of Zombies. Prior to the game, the player can customize settings including game board dimensions and number of zombies. The player can also save a game into a file and load a game from a file.

6.2. Game Board

The game is played on two-dimensional board that contains game characters and objects. No characters or objects can be placed outside the board.

The dimensions of the game board must be odd numbers so that the Alien can be placed at the centre of the game board.

For game board with 10 or more rows or columns, ensure that the row and column numbers are displayed correctly.

6.3. Game Characters

6.3.1. Attributes

Alien and Zombie are game characters. They share two common attributes: ***life*** and ***attack***. A character is defeated when its life becomes zero. The attack indicates the damage a character inflicts on its opponent's life in one hit. Unlike Alien, Zombie has the ***range*** attribute, which defines how far Zombie's attack can reach.

Before the game begins, all attributes of Alien and Zombie are initialized with a random number. The only exception is the Alien's attack which always starts from ZERO (0) at each turn. This is because Alien's attack is accumulated by collecting the Arrow objects in the game board and is reset after each turn.

To make the game interesting, the random values assigned to all attributes must be within a sensible limit. For example, attack should not be higher than life, and the Zombie's range must not be larger than any dimensions of the game board.

6.3.2. Movement and Attack

Both Alien and Zombie can move and attack, but their behaviours differ. Alien moves by continuously thrusting in one of the FOUR (4) directions (i.e., up, down, left, or right). It is stopped by one of the following events:

1. It hits the border of the game board (see Game Board).
2. It hits the Rock object (see Game Objects).
3. It hits and attacks Zombie, but Zombie survives the attack.

When Alien moves, it leaves trails on its path. At the end of Alien's turn, the trails must be reset with random non-trail game objects before the Zombie's turn begins.

Alien attacks when it encounters Zombie on the move. If Zombie is defeated by the attack, Zombie will be removed from the game board while Alien continues to move.

Unlike Alien, Zombie performs both move and attack successively at each turn. However, Zombie can only move one step in a randomly selected direction (i.e., up, down, left, or right). Zombie can only move to a location not occupied by Alien.

After a move, Zombie will attack Alien if Alien is with its attack range.

Both characters cannot move outside the board. When Alien hits a border of the board, it stops and its turn ends. Zombie must also be programmed to avoid moving outside the board.

6.3.3. Multiple Zombies

Each zombie is represented by a unique digit character in the game board starting from 1 to N, where N is the number of zombies specified by the player. The digit ZERO (0) is not used, hence the game can only support a maximum of NINE (9) zombies.

Each zombie must have different values for its attributes (i.e., life, attack, and range) which are generated randomly. You are free to decide the maximum and minimum limits of these attributes but ensure that the limits are sensible.

Zombie cannot move to locations occupied by other zombies.

When Alien finds a pod, the attack must target only **ONE (1) zombie closest to the pod**. If there are two or more zombies closest to the pod, choose any one of them at random.

For defeated zombies, the game should skip their turns, but their attributes must remain displayed throughout the game.

6.4. Game Objects

Apart from the game characters, the game board also contains game objects that Alien can interact with. Here are the game objects:

Name	Appearance	Description
Arrow	^ (up), v (down), < (left), > (right)	• Changes Alien's direction of movement. • Adds 20 attack to Alien.
Health	h	• Adds 20 life to Alien.
Pod	p	• Instantly inflicts 10 damage to Zombie when hit by Alien.

Rock	r	• Hides a game object (except Rock and Trail) beneath it. • Reveals the hidden game object when hit by Alien. • Stops the Alien from moving.
Empty	Space	• Just an empty space on the board.
Trail	.	• Left by Alien when it moves. • Reset to a random game object (except the Trail) after Alien's turn ends.

6.5. Game Controls

The player plays the game by typing commands. The following shows the commands:

Command	Description
up	Alien to move up.
down	Alien to move down.
left	Alien to move left.
right	Alien to move right.
arrow	Switch the direction of an arrow object in the game board. (The player will be asked to enter the row and column of the arrow object to switch, followed by the direction of the arrow object to switch to.)
help	List and describe the commands that the player can use in the game.
save	Save the current game to a file. (The player will be asked to enter the name of the file to save to).
load	Load a saved game from a file (The player will be asked to enter the name of the file to load from).
quit	Quit the game while still in play. (The player will be asked to confirm his/her decision)

6.5.1. Saving and Loading Game File

When saving a game, ensure that the data is stored correctly so that it can be loaded successfully. After saving, the player will continue to play the ongoing game unless the "quit" command is entered to end the game.

Before loading a game, the player must be allowed to choose whether to save or discard the ongoing game. If the player chooses to save, then the ongoing game must be saved before the loaded game is played.

You must write C++ File I/O from scratch to save and load the game data, hence you should decide how the game data is organized in the file. You are not allowed to use any existing data exchange format (e.g., JSON, XML, YAML, etc.) in which you rely on open-source parsers for the File I/O. Using these parsers takes away the opportunity to practice File I/O skills.

6.6. Game Flow

Before the game starts, the board is initialized with characters and objects. Alien always starts at the centre of the board, while Zombies and the other game objects are randomly placed in the board. The program should ensure that the characters do not overlap each other.

During the game, Alien and Zombie takes turn to act, with Alien always taking the first turn. At every turn, the board is displayed, the life and attack of both characters are updated, and messages explaining what happens at each step are showed to keep players informed about the game's progress.

The game ends when either Alien or Zombie is defeated. The player can choose to play again, and if he does, the game will restart with the board reinitialized, and the characters' life and attack reset to their initial values.

7. Additional Features

You are required to incorporate any interesting additional features to the game. You may get ideas from the original Alien Path game. Here are some quick suggestions:

7.1. New Zombie's Behaviours

You may introduce new behaviours for the Zombie's move and attack. However, the behaviours must be significantly different and harder to implement (by the examiner's standard) than those stated in the minimum requirements.

7.2. New Game Objects

You may introduce new game objects to the game. The new objects can either affects Alien, or Zombie, or both. However, the new objects must be significantly special (by the examiner's standard) than the existing game objects in the minimum requirements.

8. Deliverables

The Starter Kit that you clone to your local machine is a Git repository. Besides containing useful resources and templates to help you kick start your assignment, the repository is also the container of all the deliverables of this assignment. Therefore, you must put everything you have done for the assignment into the repository. For both Part 1 and Part 2, your submissions are the whole repository.

The following are the descriptions for each deliverable, including the expected outcomes for Part 1 and Part 2:

8.1. Source Code

Place all the source code (i.e., .cpp files, .h files) for the assignment in the repository. You are free to create new folders in the repository to organize your source code.

Start each source code with your group members' information as follows:

```
// *********************************************************
// Course: TCP1101 PROGRAMMING FUNDAMENTALS
// Year: Trimester 1, 2022/23 (T2215)
// Lab: TxxL
// Names: MEMBER_NAME_1 | MEMBER_NAME_2 | MEMBER_NAME_3
// IDs: MEMBER_ID_1 | MEMBER_ID_2 | MEMBER_ID_3
// Emails: MEMBER_EMAIL_1 | MEMBER_EMAIL_2 | MEMBER_EMAIL_3
// Phones: MEMBER_PHONE_1 | MEMBER_PHONE_2 | MEMBER_PHONE_3
// *********************************************************
```

Expected Outcomes:

A. **Part 1**: Partially completed source code contributed by all members. Progress must be significant relative to the deadline.

B. **Part 2**: Completed source codes.

8.2. Video Demonstration

For each part, you are required to record a video demonstrating the implemented assignment requirements. You must upload the video to YouTube and put the link in the Documentation.

You are free to use any tool for the video recording. If you need something enough for the job, Google Meet is your friend.

Expected Outcomes:

A. **Part 1**: A video demonstrating the implemented assignment requirements with explanation. The examiner expects to see the same output if he compiles the source code.

B. **Part 2**: A video demonstrating the complete game with explanation.

8.3. Documentations

The repository contains several Markdown files: `PART1.md`, `PART2.md`, and `README.md`. `PART1.md` and `PART2.md` is used to document your progress or work done, while `README.md` serves as your project front page when the repository is submitted to GitHub.

You are required to use these files for all documentation purposes. Please fill up these files according to the instructions written in the file.

The items you need to add in these Markdown files combined include (but not limited to):

1. **Introduction** – Introduce the game, and include the YouTube links to your video demo (see Video Demonstration)
2. **User Manual** – Provide instructions on how to navigate or play the game.
3. **Minimum Requirements** – Provide TWO (2) lists: one for completed requirements, another for to-be-done requirements (Part 1) or incomplete requirements (Part 2).
4. **Additional Features** – Describe the additional features that has been implemented, if any.
5. **Compilation Instructions** –Provide the compilation instructions if the command to compile your program is non-trivial.
6. **Contributions** – List down the contribution of each group members in detail. The examiner will verify the claimed contributions from the Git commit log.
7. **Problems Encountered and Solutions** – Describe the problems encountered when doing the assignment and provide solutions / plan for the solutions.

Expected Outcomes

A. **Part 1**: Complete `PART1.md` and update `README.md` by following the instructions in those files.
B. **Part 2**: Complete `PART2.md` and update `README.md` by following the instructions in those files.

9. Submission Instructions

Submission instructions will be posted separately later in the "**PF2223_Lecture**" Google Classroom.

10. Plagiarism

It is normal to seek help from friends or from online resources when you do the assignment, However, seeking help should not go overboard, to the point of getting (or even paying) someone to complete the assignment partly or fully for you, copying from online resources without understanding, or doing any means with the intention to cheat. For this assignment, plagiarism means the following:

A. Turning in a work that, from the examiner's point of view, you do not sufficiently understand.
B. Turning in someone else's work (whether partly or fully) as your own.
C. To use another's work (whether partly or fully) without crediting the source.
D. Any means of cheating.

Plagiarism is a serious offence.

We will give ZERO (0) marks to students who plagiarize AND to students who intentionally or unintentionally help other students to plagiarize by giving all or some of their code.

11. Evaluation Criteria

The assignment is evaluated based on the quality of the program and code. Here are the criteria in detail:

11.1. Part 1

Criteria	Description
Deliverables	
Source Code	The source code is easy-to-read and can be compiled into an executable showcasing all completed requirements.
Video Demonstration	The video clearly demonstrates every claimed implemented feature and has been uploaded to YouTube. The video link has been shared in the Markdown files as instructed.
Documentation	There is no missing or unfilled information in the documentation files.
Progress	
Correctness	The claimed implemented features and the members' contributions are proven correct (e.g., using Git Commit Logs).
Teamwork	The contributions of each group member are equal.
Rate of Progress	The amount of progress made per member relative to other groups from the same tutorial session.

11.2. Part 2

Criteria	Description
Program	
Usability and User Experience	The program's overall appearance should give a good impression to the user. The program should also be easy-to-use and provide meaningful and relevant experience to users.
Requirements	The program must fulfil all stated requirements.
Additional Features	The program must incorporate significant additional features beyond the stated requirements.
Error-free During Runtime	The program should not crash at runtime.
Code	

Efficiency	The codes should strive to use computational time and memory efficiently.
Robustness	The codes should handle unexpected cases including an accidental invalid input from the user. If the user enters the wrong input, the program should allow the user to enter another input until it is a valid response.
Modularity	The codes demonstrate significant effort in breaking down the assignment problems and solving them using functions and classes. Each function and class strive to adopt the Single Responsibility Principle. Proper use of separate compilation is a plus.
Style	The proper use of indentation, spacing, identifier naming, and more.

Please refer to the **Assignment Mark Sheet** for both Part 1 and Part 2 for detailed marks allocation of each evaluation criterion.

Assignment (Part 1) Mark Sheet

TCP1101 Programming Fundamentals
Trimester 1, Session 2022/23 (T2215)

To be filled by Examiner. This is for your reference only.

	Max	Actual Marks
1. Deliverables (40%)		
a. Source Code	20	
b. Video Demonstration	10	
c. Documentation	10	
2. Progress (60%)		
a. Correctness	20	
b. Teamwork	10	
c. Rate of Progress	30	
TOTAL (100%)	100	
Plagiarism (0 – No, 1 – Yes)	1	
FINAL MARKS (10%) = TOTAL x (1 – Plagiarism) x 0.1	**10**	

Assignment (Part 2) Mark Sheet

TCP1101 Programming Fundamentals
Trimester 1, Session 2022/23 (T2215)

To be filled by Examiner. This is for your reference only.

	Max	Actual Marks
1. Program (50%)		
NOTE: ZERO (0) mark will be given for this section if the program cannot run despite the code can be compiled.		
a. Usability & User Experience	10	
b. Minimum Requirements	25	
c. Additional Features	10	
d. Error-free During Runtime	5	
2. Code (40%)		
a. Efficiency	5	
b. Robustness	10	
c. Modularity	20	
d. Style	5	
3. Others (10%)		
a. Video Demonstration & Documentation	10	
TOTAL (100%)	100	
Plagiarism (0 – No, 1 – Yes)	1	
FINAL MARKS (30%) = TOTAL x (1 – Plagiarism) x 0.3	**30**	

- End of Assignment -